# FWBO: jQuery Edition

Clio Torrance

Scope of Work and Preplanning Document

## index.html

* Featured levels: The three most popular levels of all time and the three most popular levels uploaded this week are displayed.
* The game is playable while logged out, one just cannot create, rate, or comment on levels.
* Links >> levels.html
* Includes a JS link that takes the user to a random level. Weighted in favor of levels with no ratings.

## Header on every page:

* If logged out: Links >> login.html *and* links >> newaccount.html
* If logged in: Links >> make.html *and* links >> logout.html
  + Displays how many infractions have been incurred by the currently logged in user account.
* If logged in and moderator or admin: Links >> audit.html *and* links >> reports.html

## levels.html

* List of levels: Every level ever uploaded is displayed as a sortable, searchable list. Default sort sorts by score descending first, then upload date descending. Each level links >> play.html?id=*[level id],* so specific levels can be hotlinked from off site.
* Deleted levels: Levels deleted by moderator action are not deleted, simply flagged and inaccessible to regular users. Moderators and admins can hide them or filter for exclusively deleted levels with additional search criteria only they can see. Deleted levels show up ~~crossed out and in lighter font~~ on the list.

## play.html

* play.html with no query string returns “No level selected” and redirects >> levels.html after a short while.
* play.html?i=[valid level id] loads that level for playing, as well as the level’s title, user rating, description, and comments.
* play.html?i=[deleted level id] loads that level’s information and comments if a moderator or admin is logged in, but deleted levels cannot be played or commented on. If the user is not a moderator or admin, returns “This level has been flagged and deleted by a moderator.”
* play.html?i=[invalid level id] returns “Invalid level ID. A level with that ID has never existed.” Does not redirect, rather links >> levels.html.

## login.html

* Allows the player to log into the site. If a player is already logged in, instantly redirects >> index.html
* Allows a user who has forgotten their password to change it via email.
* When the player successfully logs in, redirects >> index.html
* Links >> newaccount.html
* Logging in creates a session. The session end time refreshes to 1 hour in the future every time a logged in user loads a page. If a user’s session is expired, they are automatically logged out next time they load a page.

## logout.html

* Logs out a logged in player.
* Redirects >> index.html

## newaccount.html

* Allows the player to create an account, requiring an email, a username, and a password. The email is exclusively used for account retrieval in the case of a forgotten password, and must be verified before the account can be used.
* If already logged into an account, instantly redirects >> index.html

## make.html

* Allows logged in users to create levels.
* If a user is not logged in, returns “Only logged in users can create levels.” and links >> login.html *and* links >> newaccount.html
* Shows a grid. The player can select from a variety of objects and click to place them on the grid.
* Clicking to place the player spawn point or the exit deletes any other existing instances of those first.
* When a player spawn point and an exit are placed, links >> verify.html
* Uses cookies to store the current level being made so it is preserved through internet or server outages and page changes. There is a “clear” button on make.html to remove this cookie and start over.

## verify.html

* A user is required to test their level before it can be stored on the server. It must be completable.
* Loads a playable version of the cookie stored level.
* Links >> make.html so the user can go back to editing their level if they’ve accidentally made it unplayable.
* Once the player successfully reaches the end, links >> submit.html
* If not logged in, redirects >> levels.html
* If logged in but there is no level cookie, redirects >> make.html

## submit.html

* Once a user has created and verified their level, they can name, describe, and submit it.
* Name is required. It doesn’t have to be unique, but a warning will be displayed if a level already exists with that name.
* Description is not required and can be left blank.
* If not logged in, redirects >> levels.html
* If logged in but there is no completed level cookie, redirects >> make.html
* How I plan to have the cookie stores whether the level is complete in an untamperable way:
  + When verify.html links to submit.html, the link first executes Javascript that adds the UNIX time that the link was clicked to the cookie.
  + If submit.html loads within X seconds (reasonable loading time but ideally too fast for the cookie to be edited) of said time, it is valid. If not, the player is prompted to re-verify the level.
  + submit.html being very light weight is supposed to help loading times and minimize the people this verification method bars from submitting their levels.

## audit.html

* Loads entries from the modactions table, a new entry for which is added each time a moderator or admin takes an action in their official capacity.
* Sortable and searchable. Intended to promote accountability and transparency among the mod team. Default sort order is reverse chronological order.
* If not logged in or not a mod/admin, redirects >> index.html instantly.

## reports.html

* Loads entries from the reports table. Users can report levels and comments for being offensive or inappropriate. Each user can report any given level or comment only once.
* Sortable and searchable. Default sort order is reverse chronological.
* Mods can take action from this page or view the reported content in more detail.
* If not logged in or not a mod/admin, redirects >> index.html instantly.

## Moderator Privileges:

* Delete a level from public accessibility. Doing so requires a description of why, which will be shown to the level’s creator next time they load a page.
* Delete a comment. Doing so requires a description of why, which will be shown to the commenter next time they load a page.
* Send a private warning message to a user or moderator.
* Bar a user from commenting for one week.
* Bar a user from submitting levels for one week.
* Apply an infraction to a user account along with any above action if the violation is deemed serious enough.
* Permanently ban a user account. Requires three or more infractions on said account.
* Username displayed in bold purple.

Usernames of banned accounts are displayed as *~~Banned User~~* instead of the actual name in case profanity in the username was the reason for banning.

## Admin Privileges (in addition to the above):

* Promote a user to a moderator.
* Demote a moderator to a regular user.
* Permanently ban a user or moderator account at will.
* Username displayed in bold italic orange.

Only someone with direct access to the SQL table can promote a user to admin.

## Features I’m Considering

* Curator status
  + Can only rate 1 level per day, and can only rate 5 stars, but ratings carry 10x the weight.
  + Regular users can apply for the status and moderators can allow or deny that based on their past behavior.
  + Username displayed in green or blue or something.
  + Curator page showing what levels that user has rated?
* User audit pages (audit.html?u=[id])
  + See every level and comment the user has ever submitted in reverse chronological order, their join date, and infraction count. Also shows modactions taken by that user if they are a mod or admin.
  + Could also just implement a user page that would show created levels and submitted comments, also displaying audit information to logged in mods and admins.
* (more to be added as warranted)

## SQL tables

* **user**
  + **user\_id** int; primary key
  + **join\_date** datetime
  + **email** string
  + **is\_verified** Boolean
  + **last\_login** datetime
  + **username** string; 4-20 characters
  + **pw\_hash** string
  + **infractions** unsigned byte (Don’t let a user get to 255 infractions before banning them please)
  + **privs** (255)admin, (10)moderator, ((5)curator,) (0)permabanned, or (anything else)none
  + **level\_ban\_expires** datetime
  + **comment\_ban\_expires** datetime
* **level**
  + **level\_id** int; primary key
  + **submit\_date** datetime
  + **by\_user\_id** int; links to user.user\_id
  + **name** string; 5-40 characters
  + **desc** string; 0-200 characters
  + **content** string with 1 character for every grid square in the level with its object id
  + **is\_deleted** Boolean
* **rating**
  + **rating\_id** int; primary key
  + **rating\_date** datetime
  + **by\_user\_id** int; links to user.user\_id
  + **on\_level\_id** int; links to level.level\_id
  + **is\_curator** Boolean
  + **score** 1-5
* **comment**
  + **comment\_id** int; primary key
  + **comment\_date** datetime
  + **by\_user\_id** int; links to user.user\_id
  + **on\_level\_id** int; links to level.level\_id
  + **content** string; 0-150 characters
* **modaction**
  + **action\_id** int; primary key
  + **by\_user** int; links to user.user\_id of moderator
  + **on\_user** int; links to user.user\_id of action’s target
  + **action\_date** datetime
  + **action\_type** delete\_level, delete\_comment, level\_ban, comment\_ban, mod\_promote, mod\_demote, or warning
  + **involves\_level** int; links to level.level\_id
  + **involves\_comment** int; links to comment.comment\_id
  + **action\_reason** string; 20-200 characters
  + **incurs\_infraction** Boolean
* **report**
  + **report\_id** int; primary key
  + **report\_date** datetime
  + **reporting\_user\_id** int; links to user.user\_id
  + **reported\_user\_id** int; links to user.user\_id
  + **concerns\_level\_id** int; links to level.level\_id
  + **concerns\_comment\_id** int; links to comment.comment\_id
  + **report\_reason** string; 20-200 characters
  + **acknowledged\_by\_mod** Boolean; defaults to false
* **registration\_token** solely used for email verification - removed once user is verified
  + **token\_id** randomly generated unique 20-alphanumeric-character string
  + **for\_user\_id** int; links to user.user\_id
* **session**
  + **session\_id** randomly generated unique 20-alphanumeric-character string
  + **for\_user\_id** int; links to user.user\_id
  + **expires** datetime